General Linear Least Squares

James R. Graham
2014/10/6

Equations of condition

Suppose we consider a model to describe a data set (x_i, y_i) where $y = y(x)$ and the function can be written in the form

$$y_i = \alpha_1 \beta_1(x_i) + \alpha_2 \beta_2(x_i) + \cdots \alpha_n \beta_n(x_i) + e_i \;, \tag{1}$$

where β is some known function of the independent variable x_i, α_i are constants, and the unknown measurement errors, e_i. If the problem can be expressed in this manner it is a linear one, because the dependent variable is a linear combination of known functions of the independent variable. If we write

$$B_{ij} = \beta_j(x_i)\,, \tag{2}$$

and

$$\mathbf{a} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}, \tag{3}$$

then we can write the problem in matrix form as

$$\mathbf{y} = \mathbf{B}\mathbf{a} + \mathbf{e}\,. \tag{4}$$

The equations represented by Eq. (4) are known as the ***equations of condition***. The individual measurement errors are unknown—our goal is to find **a**, gven the data, that minimizes the inferred errors, i.e., that minimizes **y**-**Ba**. The quantity **y**-**Ba** is a matrix (1 column by n rows). What do we mean by minimizing a matrix?

Least squares

The notation $\|\ldots\|_2$ is used to denote the Euclidian vector norm,

$$\|\mathbf{x}\|_2^2 = \mathbf{x}^T\mathbf{x} = \sum_i x_i^2 \;, \tag{5}$$

where the superscript T denotes the transpose. If we think of the Pythagorean theorem in n dimensions, the Euclidian vector norm is a measure of the length of the vector **x**. Using the norm notation of Eq. (5) we can write a compact expression for the sum of the squares of the residuals,

$$\chi^2 = \|\mathbf{y} - \mathbf{Ba}\|_2^2 . \tag{6}$$

Expanding, we find

$$\begin{aligned} \chi^2 &= (\mathbf{y} - \mathbf{Ba})^T (\mathbf{y} - \mathbf{Ba}) \\ &= \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{Ba} + \mathbf{a}^T \mathbf{B}^T \mathbf{Ba}, \end{aligned} \tag{7}$$

by using the distributive property of the transpose and $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$.

We want to minimize this expression—or the *least squares*—as a function of **a**, so that the first derivatives[1] with respect to **a** are zero

$$\frac{\partial \chi^2}{\partial \mathbf{a}} = -2\mathbf{B}^T \mathbf{y} + 2\mathbf{B}^T \mathbf{Ba} = 0 , \tag{8}$$

or

$$\mathbf{B}^T \mathbf{Ba} = \mathbf{B}^T \mathbf{y} . \tag{9}$$

Thus, the unknown vector **a** is found by multiplying each side by the inverse matrix $(\mathbf{B}^T\mathbf{B})^{-1}$

$$\begin{aligned} \left(\mathbf{B}^T \mathbf{B}\right)^{-1} \mathbf{B}^T \mathbf{Ba} &= \left(\mathbf{B}^T \mathbf{B}\right)^{-1} \mathbf{B}^T \mathbf{y} \\ \mathbf{a} &= \left(\mathbf{B}^T \mathbf{B}\right)^{-1} \mathbf{B}^T \mathbf{y}. \end{aligned} \tag{10}$$

The quantity $(\mathbf{B}^T\mathbf{B})^{-1}$ is known as the generalized or Moore-Penrose pseudo-inverse of **B**. Sophisticated versions of general least squares methods use ***singular value decomposition*** to compute the inverse of $\mathbf{B}^T\mathbf{B}$.

If the measurement error is not constant or measurements are not independent but covariant, then Eq. (7) is replaced by

$$\chi^2 = (\mathbf{y} - \mathbf{Ba})^T \mathbf{V}^{-1} (\mathbf{y} - \mathbf{Ba}) \tag{11}$$

where the weights are the inverse of the covariance matrix **V.** Under these circumstances, the least squares parameters are now

---

[1] See “The Matrix Cookbook”, 2006 Petersen & Pedersen, MIT

$$\mathbf{a} = \mathbf{H}\mathbf{y} \tag{12}$$

where

$$\mathbf{H} = \left(\mathbf{B}^T\mathbf{V}^{-1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{V}^{-1} . \tag{13}$$

Error propagation yields estimates of the covariance matrix of the parameters, **a**, (and the variance of these parameters as the diagonal matrix elements) from

$$\mathbf{C} = \mathbf{H}\mathbf{V}\mathbf{H}^T = \left(\mathbf{B}^T\mathbf{V}^{-1}\mathbf{B}\right)^{-1} . \tag{14}$$

Example 1: Uniform acceleration from rest

Suppose we have a set of data described by a parabolic relation

$$x = \frac{1}{2}gt^2 ,$$

e.g., the distance traveled by a body dropped from rest at time zero. How do we find the value of g? Some data are shown in Figure 1.

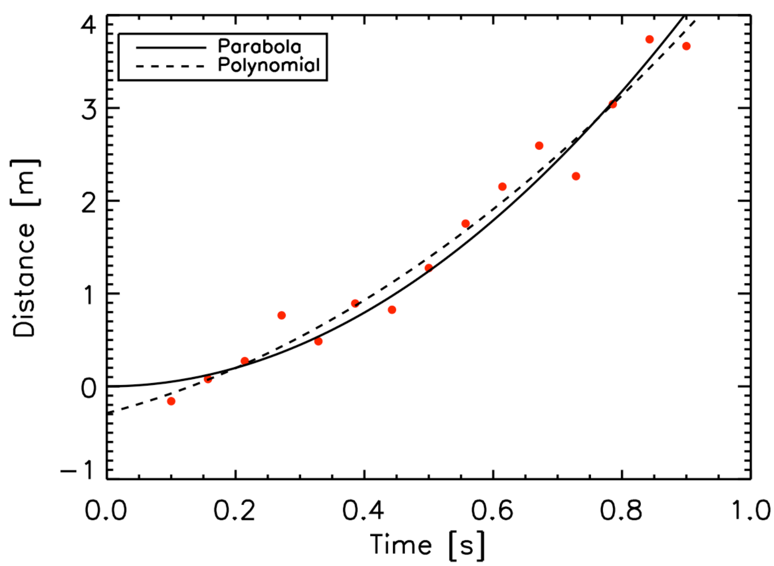

Figure 1: Measurement of the position of a body falling from rest under gravity with g = 9.81 m s^{-2}. The dotted line shows the fit that you get if you fit a general quadratic.

Writing the measurement in the form of a matrix equation

$$\underbrace{\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} \frac{1}{2}t_0^2 \\ \frac{1}{2}t_1^2 \\ \vdots \\ \frac{1}{2}t_{n-1}^2 \end{pmatrix}}_{\mathbf{B}} (g) + \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_{n-1} \end{pmatrix},$$

we can identify the matrices **y** and **B** by comparison with Eq. (4). If the data vectors are `t` and `x`, in Python the solution is implemented as follows:

```
import numpy as np
.
.
.
b = np.transpose(np.matrix(0.5 * t**2)) # column matrix-independent variable
y = np.transpose(np.matrix(x))          # column matrix-dependent variable

bt = np.transpose(b)
btb = bt * b
mpsi  = np.linalg.inv(btb)                # compute MP pseudo inverse
g = mpsi * bt * y
```

Note that the `matrix` and `transpose` in the first two steps convert arrays into a 1-column by n-rows matrix. The objects `b`, `y`, `bt`, `btb`, and `mpsi` are matrices, so the operator * performs conventional matrix multiplication.

The conventional (but wrong) approach would be to fit a second order polynomial to the data. In the example in Figure 1, the parabolic fit gives $g = 9.93$ m s^{-2}, whereas the polynomial fit implies that the initial position is -0.29 m, the initial velocity is 1.81 m s^{-1} and the acceleration is 6.18 m s^{-2}. Polynomial fitting fails to take account of our knowledge that the initial position and velocity are zero, and as a consequence gives an inaccurate value for the acceleration.

Example 2: Uniform circular motion

Now suppose our task is to determine the radius of a wheel by measuring the x-coordinate of a point on the circumference as the wheel rotates at a known frequency ω. The position of that point is given by

$$x = x_0 + R\sin(\omega t)\ .$$

From measurements of (t, x) we want to find x_0 and R. For this example, the relevant fragment of code is

```
import numpy as np
.
.
.
b1 = np.ones(npts)                        # Independent variable
b2 = np.sin(omega*t)
```

```
b = np.matrix(np.column_stack((b1,b2)))
bt = np.transpose(b)

y = np.transpose(np.matrix(xe))          # Dependent variable

btb = bt * b                             # Compute MP pseudo inverse
mpsi  = np.linalg.inv(btb)

ans = mpsi * bt * y                      # Find the least squares solution
```

An example of such a fit is shown in Figure 2.

Note the limitation of this method—we cannot determine ω from the data; we have to know the rotation rate. Problems where unknowns enter other than in linear combinations fall into the category of ***non-linear least squares***. There are no closed-form solutions to non-linear problems: they are solved using iterative methods that require an initial guess for the model parameters.

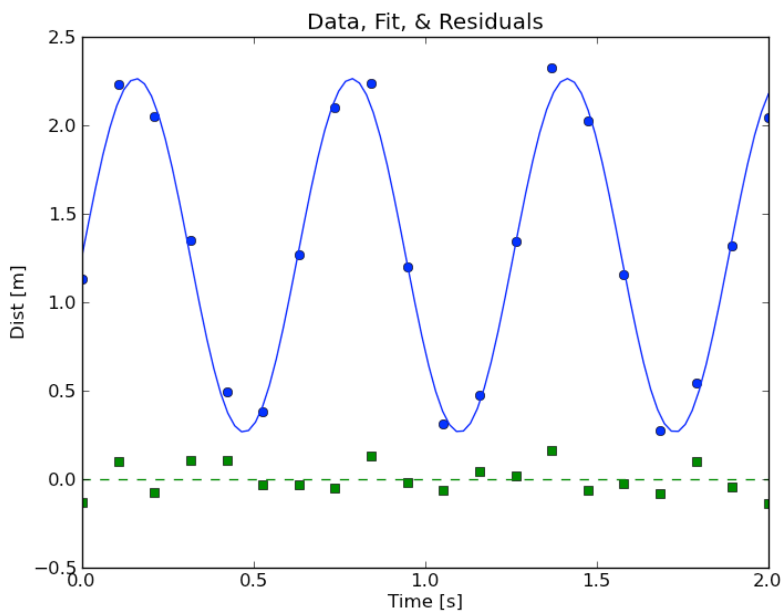

Figure 2: Time series of measurement of a point on the circumference of a wheel rotating at known angular frequency ω. A linear least squares fit to $x = x_0 + R\sin(\omega t)$ yields the radius and the x-coordinate of the point of rotation. The residuals between the data (blue squares) and the fit (blue line) are shown as green squares.

At first sight some problems appear non-linear, e.g., the case of the rotating wheel when the phase, ϕ, is unknown

$$x = x_0 + R\sin(\omega t + \phi) .$$

However, by use of trigonometric identities we can write

$$x = x_0 + R\cos(\phi)\sin(\omega t) + R\sin(\phi)\cos(\omega t) \, ,$$

which is a linear problem.